Subqueries

A brief explanation

What is a Subquery?

A Subquery is a SELECT expression that is embedded in a clause of SELECT statement to for a final statement.

We use subqueries to perform complex comparisons, generate and display data. Whether it is cycling through recipes to find all with a particular ingredient or checking client’s most recent purchases subqueries enable us to perform these operation efficiently.

3 Types of Subqueries

- Row subquery
  - An embedded SELECT statement that returns one or more columns but no more than one row.
- Table subquery
  - An embedded SELECT statement that returns one or more columns and one to many rows.
- Scalar subquery
  - An embedded SELECT statement that returns one column and no more than one row.

Table Subquery example

Does this look familiar?

We’ve been doing Table subqueries for quite a while now.

```
SELECT Customers.CustLastName
FROM ((Customers INNER JOIN Orders ON Customers.CustomerID=Orders.CustomerID)
INNER JOIN Order_Details ON Orders.OrderNumber=Order_Details.OrderNumber)
inner JOIN Products ON Products.ProductNumber=Order_Details.ProductNumber
WHERE Products.ProductName LIKE '% helmet'

UNION

SELECT Vendors.VendName
FROM (Vendors INNER JOIN Product_Vendors ON
Vendors.VendorID=Product_Vendors.VendorID)
INNER JOIN Products ON Products.ProductNumber=Product_Vendors.ProductNumber
WHERE Products.ProductNumber LIKE'% helmet'
```

Scalar Subquery example

```
SELECT Orders.OrderNumber, Orders.OrderDate, Orders.ShipDate,

(SELECT Customers.CustLastName FROM Customers WHERE
Customers.CustomerID = Orders.CustomerID)  AS CustomerName

FROM Orders
WHERE Orders.ShipDate='2012-10-03'
```

Aggregate Functions

Aggregate Functions allow you to calculate a single value from the rows in a result set or values returned by an expression.

For example finding the average length of a class or counting the number of clients who live in Seattle.

COUNT()

COUNT(*) - which is shorthand for Count all - is used to find out how many rows are in an entire set.

COUNT(Column_Name) - will count all rows in the column with Non-Null values.

MAX()

MAX returns the highest or most recent value of a column.

If the value expression is numeric it will return the highest numbers. However if the value is date or time related MAX will return the most recent date or time value.

Aggregate Functions - COUNT()

What if I asked you to show me all of the customer names and any orders they may have placed.

Run this query and examine the results.

```
SELECT Customers.CustFirstName,
Customers.CustLastName, (SELECT COUNT(*) FROM Orders
WHERE Orders.CustomerID=Customers.CustomerID) AS
CountOfOrders FROM Customers
```

Aggregate Functions - MAX()

This tables displays the customer's full name and the last date on which they placed an order.

```
SELECT Customers.CustFirstName,
Customers.CustLastName, (SELECT MAX(OrderDate)
FROM Orders  WHERE
Orders.CustomerID=Customers.CustomerID) AS
LastOrderDate
FROM Customers
```

**Don't forget when you use MAX in relation to a date it will return the most recent date.*

Quantified Predicate Keywords

As its name would imply when you use the keyword “**ALL**” then the comparison must be true for all values returned by the subquery.

Let’s explore this example!

```
SELECT Products.ProductName, Products.RetailPrice
FROM Products
INNER JOIN Categories
ON Products.CategoryID=Categories.CategoryID
WHERE Categories.CategoryDescription='Accessories'
AND
Products.RetailPrice<ALL
(SELECT Products.RetailPrice
FROM Products
INNER JOIN Categories
ON Products.CategoryID=Categories.CategoryID
WHERE Categories.CategoryDescription='Clothing')
```

Quantified Predicate Keywords

SOME and ANY

When you use the keywords “SOME” or “ANY” then the comparison has to only be true for at least one value in the list.

Please note *SQL Standard treats these two keywords <u>ALL</u> and <u>SOME</u> as equivalents.

```
SELECT Recipes.RecipeTitle
FROM Recipes
WHERE Recipes.RecipeID IN
(SELECT Recipe_Ingredients.RecipeID
FROM Recipe_Ingredients
WHERE Recipe_Ingredients.IngredientID=SOME
(SELECT Ingredients.IngredientID
FROM Ingredients
WHERE Ingredients.IngredientName
IN ('Chicken','Garlic')))
```

**Try this again and replace the keyword “SOME” with keyword “ANY”. What do you notice? See the bold text to the left for an explanation.*

Exists

This is useful when you want to check if a related row “exists”.

For example if you want to know if any customers have purchased any clothing.

```
SELECT Customers.CustFirstName + ' '+Customers.CustLastName AS
[CustName]FROM Customers
WHERE EXISTS
(SELECT *
FROM (ORDERS
INNER JOIN Order_Details
ON Orders.OrderNumber=Order_Details.OrderNumber)
INNER JOIN Products
ON Products.ProductNumber=Order_Details.ProductNumber
WHERE Products.CategoryID=3
AND Orders.CustomerID=Customers.CustomerID)
```

Try It-1 School Scheduling db

List all staff members and a count of classes each teaches.

Try It-2 Entertainment Agency db

List entertainers who played engagements for customers Berg or Hallmark.

Try It-3 Entertainment Agency db

Display Agents who haven’t book an entertainer.